Claims 1-23 have new grounds of rejection under 103(a) and the response herein is with respect to these rejections.

Claims 24-33 stand rejected from the First Office Action dated February 1, 2000, as the arguments provided in the response to the First Office Action were deemed not to be persuasive. In the first office action Claims 26-27 were rejected under 35 U.S.C. 112 and 103, Claims 24-25 were rejected under 35 U.S.C. 112 and 102. Claims 28-33 where rejected under 35 U.S.C. 102.

Per the Examiner's request, Figure 1 will be corrected to reflect that this figure is Prior Art. Applicants will submit the Formal Drawings at a later date. No new matter has been added.

September 11, 2000 Arguments

35 U.S.C. §103(a)

Claims 1-14, 16-18, 20, 22, 23

In the September 11, 2000 Office Action, claims 1-14, 16-18, and 20[1] were rejected under 35 U.S.C. § 103(a) as being anticipated by Chen et al. "A Source-Level Dynamic Analysis Methodology and Tool for High-Level Synthesis", Proceedings of the Tenth International Symposium on System Synthesis, 1997, pp. 134-140 (hereinafter

---

[1] While item 5 in the office action states that 1-14, 16-18 and 20 are rejected under 35 U.S.C. 103(a), arguments were provided later in that section for additional claims 22 and 23.

"Chen") further in view of Kucukcakar et al. "Matisse: an architectural design tool for commodity ICs" (hereinafter "Kucukcakar") and Fang et al. "A Real-time RTL Engineering-change Method Supporting On-line Debugging for Logic-emulation Applications" (hereinafter "Fang").

Claim 1 reads

> **A method comprising the steps of:**
> **a) identifying at least one statement within a register transfer level (RTL) synthesizable source code; and**
> **b) synthesizing the source code into a gate-level netlist including at least one instrumentation signal, wherein the instrumentation signal is indicative of an execution status of the at least one statement.**

Chen, in contrast, teaches a system that enables HLS (High Level Synthesis) to offer source-level design debugging on 'synthesized' RTL designs. 'Synthesized' RTL designs are RTL representations of designs that have been synthesized (via High Level Synthesis or HLS) from the behavioral description. In other words, in Chen an HLS generated design is a 'synthesized' RTL design rather than a "gate-level netlist" as claimed in claim 1.

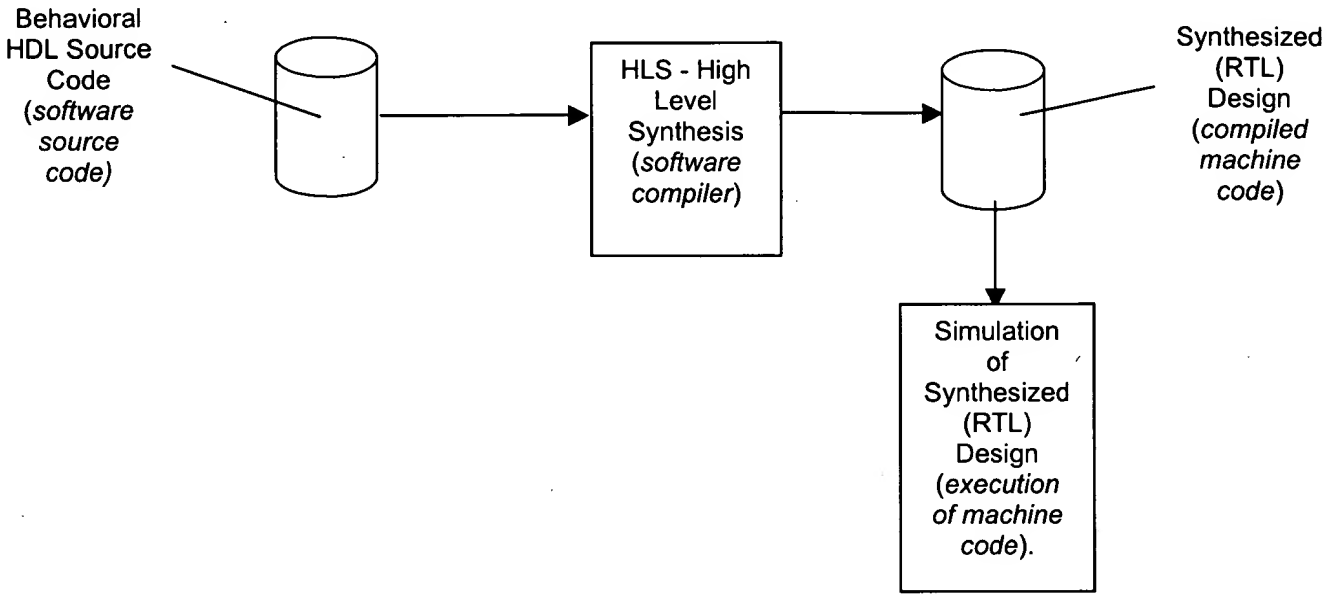
Specifically, the 1st paragraph of the introduction in Chen discusses two common problems associated with traditional HLS tools. Beginning with the 5th sentence. "Two specific problems among [traditional HLS tools] are the controllability/predictability and simulation/analysis of the synthesized RTL implementations." Next the author states that "The predictability is found to be crucial for subsequent design tasks including RTL/logic synthesis...".

To be a gate level design the design would have to have already gone through the task of RTL/logic synthesis. Therefore it does not logically follow that, if a synthesized RTL implementation is a gate-level design, a subsequent task to be performed would be RTL/logic synthesis. Resultantly Applicant respectfully submits that the HLS process discussed throughout Chen including referenced sections 3 and 3.1, and provided by the Matisse environment, does not contemplate the process of generating a gate-level netlist with design annotation, but rather is directed merely to an RTL implementation with annotated signals.

The analogy provided by Chen states

> A portion of this methodology is analogous to the source-level debugging of a compiled software code, where the programmer debugs the execution of the compiled machine code while viewing the original source code. In this analogy, **HLS,** *the input description,* and the simulation of the synthesized design correspond to **the software compiler**, *the software source code* and the execution of the compiled machine code.

The following figure shows the analogy in graphical form:



In the analogy, HLS (and not RTL/logic synthesis as discussed above) is shown as the equivalent of a software compiler. This analogy does not include an RTL/Logic synthesis step. This RTL/Logic synthesis step would be required if the generated "compiled machine code" was to be a gate-level design. Resultantly, the synthesized design is a synthesized RTL design as previously discussed. Therefore the resulting simulation of synthesized design is the simulation of the synthesized RTL code, **not** the simulation of the gate level representation.

Thus, Chen does not suggest or disclose identifying RTL synthesizable code statements and synthesizing the code into a gate-level netlist including at least one instrumentation signal as claimed in claim 1.

The Final Office Action refers to Kucukcakar and the section "Dynamic Execution Analysis" (DEA) along with Figure 7 as support for the contention that Matisse contemplates instrumentation of gate-level designs. However, the Dynamic Execution Analysis section discusses an architectural design process that doesn't contemplate gate level design generation. The first paragraph of the DEA section states "The implementation resulting from the architectural design process must be operational in a larger system." From this we can gather

1. that there is a *process* and it is an *architectural design process* and
2. the output (e.g. result) from this process is an *implementation*.

Further in this paragraph it is stated "Using standard HDL debugging tools on the architectural design output (the RTL design) ...". Thus the architectural design process:

- ✓ produces an output and
- ✓ this output is the RTL design implementation, not a gate level design implementation.

Thus, Matisse works only with RTL output, not the gate-level output.

The schematic shown in Figure 7 (part c) is not a schematic of a gate level representation (e.g. circuit) for the design. Figure 7(c) shows a schematic of the structure of the design. As noted in the last paragraph of the Dynamic Execution Analysis section, the "structural profile" is merely profile information on the algorithm specification (e.g. behavioral HDL). Moreover, if this were an actual schematic of a gate level representation, there would be logic flowing from the reset signal and control logic for select lines to the muxes.

For the aforementioned reasons, applicant respectfully submits that Matisse, and the disclosures of Chen and Kucukcakar do not teach instrumenting gate-level designs as claimed in claim 1. Rather, Chen and Kucukcakar discuss the annotation of behavioral level designs such that, after HLS is run on the behavioral level designs, the resulting RTL designs can be simulated such that source level (behavioral) execution analysis can be performed.

Fang is cited for a system whereby RTL changes can quickly be debugged (section 3.3, page 104). However, Fang is directed toward solving the problem of being able to determine where, in the HDL source code, the logic for a particular Configurable Logic Block (CLB) is located (abstract and section 1). This is to enable the debugging of designs that are being emulated in a system, wherein the emulator is designed using FPGAs. Moreover, the method involves developing *HDL structure trees* to solve the problem described therein. Thus, Fang does not suggest or disclose *instrumentating HDL source* code as disclosed in claim 1.

Finally, even if Fang did teach instrumenting RTL for emulation it would not be reasonable to combine the Fang and Hsu references to infer obviousness of instrumenting RTL for simulation. "It is improper to combine references where the references teach away from their combination." MPEP §2145 (X)(D)(2). "A prior art reference must be considered in its entirety, i.e., as a whole, including portions that would lead away from the claimed invention." MPEP §2142.02.

Chen teaches away from instrumenting RTL for simulation by stating that "the design annotation should be performed such that the analyzable HLS design doesn't

incur any overhead in the downstream synthesis tools as compared to the original HLS design." By the very nature of the present invention, the design annotations must be performed such that overhead occurs during the logic synthesis process. That is, the resulting gate level netlist from the logic synthesis process must have the instrumentation logic required for the gate level simulation of the present invention to work.

For at least the reasons stated above, applicant respectfully submits that Claim 1 is not obvious over Chen further in view of Kucukcakar and Fang. Given that claims 5, 12, 16 and 20 include substantially equivalent elements to that of claim 1, Applicants respectfully submit that claims 5, 12 16 and 20 are similarly not obvious over Chen further in view of Kucukcakar and Fang. Given that claims 2-4, 6-11, 13-14, 17-18 and 22-23 depend from allowable claims 1, 5, 12, 16 and 20 respectively, applicant respectfully submits that these claims are also not obvious over Chen further in view of Kucukcakar and Fang.

February 01, 2000 Arguments

35 U.S.C. 112

Claims 24-27

In the February 01, 2000 Office Action, claims 24-27 were rejected under 35 U.S.C. 112, second paragraph. In the Final Office action dated September 11, 2000, the rejections of claims 24-27 under 35 U.S.C. 112 in the previous Office Action were maintained and the arguments with respect to this rejection argued in the Applicants response of 6/1/00 were deemed not to be persuasive.

The Office Action rejects claims 24-27 for failure to claim the subject matter which the Applicant regards as the invention. Specifically, the Office Action rejects the process of identifying a sensitivity list by stating that it is unclear from the specification whether the sensitivity list is automatically generated, or if this list is generated manually as the code is modified by the insertion of the instrumentation logic.

Figure 18 is illustrative of the process shown in Figure 17. As mentioned in the specification, the italicized code is the code that is added by an embodiment of the present invention (page 25, line 18 - 1810 and 1820). The non-italicized code is, therefore, an exemplary existing fragment of code from a VHDL source file that is read by the present invention. Note that, as mentioned in the specification, the sensitivity list of a Verilog "always" statement will have similar analysis.

In the VHDL case, we specifically have

P1: PROCESS(A,B,C)

which is the portion of code that would pre-exist as part of the user design. The embodiment of the invention will read the source file containing the process whose name is P1 and encounter the keyword PROCESS ("P1" is an optional label of the process). After this, per the VHDL language reference specification, the embodiment of the present invention will look for the "(" identifying the beginning of the sensitivity list. As soon as the "(" is encountered, subsequent identifiers, prior to the terminating ")", are signals that constitute the sensitivity list. Resultantly, during the task of parsing a VHDL source file, containing the aforementioned portion of code, for simulation, this embodiment will encounter the signals A, B and C which are the signals comprising the sensitivity list.

After determining the set of signals in the sensitivity list for process P1, the embodiment of the present invention will add the code shown in Figure 18 (1810 and 1820). The added "process" statement (1810) and "concurrent signal assignment" statement (1820) combine to provide the logic to identify differences in sensitivity list signal values between simulation cycles. The process statement that is added will operate to produce "memorized" versions of the signals in the sensitivity list. This will happen when FAST_CLK is toggled (FAST_CLK is described in detail on page 21 lines 19-23). When subsequent activity occurs on any of the signals in the sensitivity list of P1, this will trigger the concurrent assignment statement (1820, this is because a signal assignment statement is sensitive to changes on ANY signal to the right of the "<=" operator) that will cause one of the three conditions to be true. This is because one of A, B or C will be different from the sampled values.

To summarize, the sensitivity list for which Figure 17 refers will be a sensitivity list for either a VHDL "process" or Verilog "always" statement (page 24, lines 18-20) that preexists and is to be simulated (the non-italicized code, in this embodiment as described on page 25 and as shown in figure 18). During the task of performing the instrumentation of the simulation of the process (or "always" block) in one embodiment, code may be added that will itself contain a process (or "always") statement (italicized code in figure 18). This process will not be instrumentated and is code that will be added (the italicized code is added, in this embodiment as described on page 25 and as shown in figure 18).

Thus applicants respectfully submits that "identifying a sensitivity list of a process;" of claims 24-27 does claim subject matter of the present invention and is therefore allowable under 35 U.S.C. 112, 2nd paragraph.

35 U.S.C. 102

In the February 01, 2000 Office Action, claims 24-25 and 28-33 were rejected under 35 U.S.C. 102(b). In the Final Office action dated September 11, 2000, the rejections of the previous Office Action were maintained and the arguments with respect to this rejection as argued in the Applicants response of 6/1/00 were deemed not to be persuasive.

Claims 24 and 25

Claim 24 was rejected as being anticipated by Chen. The Office Action states that Chen teaches the use of a combination of Execution Database and Value Change

Database to collect information from embedded daemons after simulation is run. Further, it is stated that the iterative design environment would allow for the use of these databases to be reviewed and used as the basis for design changes. However, as discussed above, Chen discloses a method for simulating RTL designs generated as part of a High Level Synthesis process wherein the RTL designs are annotated. In contrast, claim 24 operates as a method of simulating a **gate-level** design. Resultantly, Chen does not anticipate claim 24. Since claim 25 depends from claim 24, applicant respectfully submits that claim 25 is also not anticipated by Chen.

Claims 28-33

Chen discloses annotating designs through an HLS process, resulting in RTL implementation designs. However, as discussed above, these designs are not gate level designs and as a result do not apply to the processes disclosed in claims 1, 5, 12, 16 and 20. Claims 28-33 are machine readable medium claims that contain substantially the same limitations as the aforementioned claims. Resultantly these claims are also not anticipated by Chen.

In conclusion, Applicant respectfully submits that claims 1-33 are in a condition for allowance, and Applicant respectfully requests allowance of such claims.

Respectfully submitted,

COLUMBIA IP LAW GROUP, LLC

Dated: 11/9, 2000

Robert A. Diehl
Registration No. 40,992

4900 SW Meadows Road, Suite 109
Lake Oswego, Oregon 97035
503-534-2800